Laboratorul 05.

Arhiva laborator 5: arhiva5.zip

Problema 1

Corectati erorile, daca exista, din fisierul ***Problema1***, existent in arhiva laboratorului si verificati care este rezultatul rularii clasei ***Problema1***, incercand sa-l intuiti prima data si apoi sa intelegeti de ce este asa, daca nu este cel la care v-ati fi asteptat.

Modificati clasele puse la dispozitie in fisierul ***Problema***, din arhiva laboratorului, astfel incat rezultatul rularii clasei ***Problema1*** sa devina cel de mai jos. Puteti modifica **doar** tipul sau ordinea blocurilor de initializare!

```
Bloc 2 - Animal
Bloc 2 - Caine
Bloc 1 - SharPei
Bloc 2 - SharPei China
Bloc 1 - Animal
Bloc 1 - Caine negru
Bloc 4 - Caine
Bloc 3 - Caine
Bloc 3 - SharPei 5
```

Problema 2

Pornind de la clasa ***MyList***, oferita in arhiva laboratorului, implementati clasa derivata ***Graph*** care defineste un graf orientat, fara costuri asociate, reprezentat prin liste de adiacenta. Graful va avea nodurile numerotate incepand cu 1, iar clasa ***Graph*** va avea un constructor care primeste un numar intreg ca argument, reprezentand numarul de noduri din graf. Clasa va contine si un vector vizitat ce va fi folosit la parcurgearea in adancime.

- **Metodele clasei:**

1. `void add(int x, int y)` = adauga arcul (x, y) in graf;
2. `void dfs(int start)` = realizeaza parcurgerea in adancime a grafului, pornind din nodul ***start*** si afisand fiecare nod vizitat;
3. `String toString()` = afiseaza graful sub forma:

```
nr nod: lista de noduri vecine (pe cate o linie)
```

Pentru testare, se poate folosi clasa executabila ***TestGraph***, din arhiva laboratorului, in care se defineste graful din figura de mai jos, aplicandu-se o parcurgere in adancime pornind din nodul **1**.

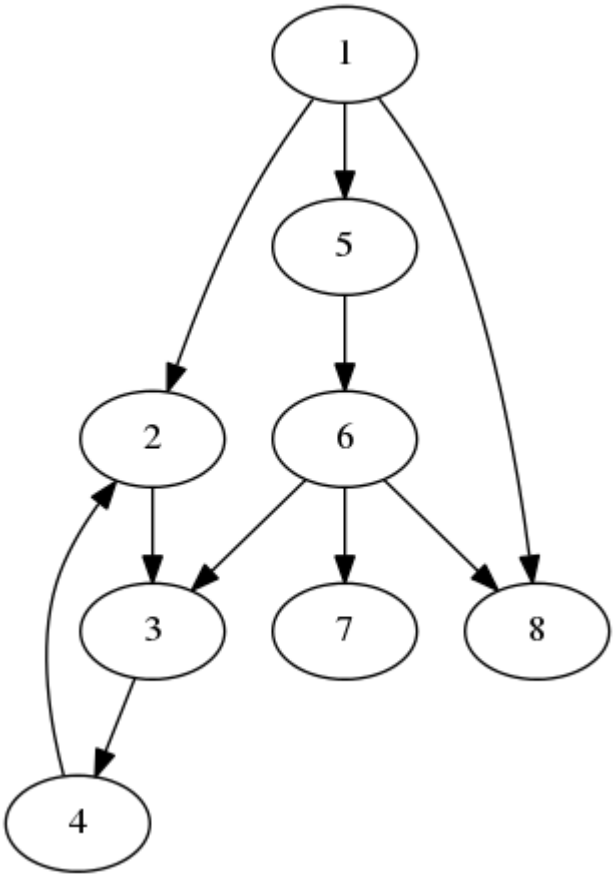
```
MyList list = (MyList) get(x);
```

Metoda ***dfs*** se poate implementa recursiv si, in acest mod, nu mai este nevoie de uzitarea explicita a unei structuri de date auxiliare.

Pseodocod pentru dfs:

```
dfs(i) {
    ia lista l a nodurilor vecine nodului i
    afiseaza nod i
    seteaza  vizitat[i] = 1;
    daca (list != null) {
        pentru fiecare nod j din lista vecinilor lui i
            daca j nu a fost vizitat
                dfs(j)
```

```
    }
}
```




```
1 2 3 4 5 6 7 8
```

Problema 3

Sa se scrie un program care citeste un sir de numere intregi de la tastura, pana la intalnirea unui numar negativ, si determina elementul maxim din sir. In implementare, se va utiliza un obiect de tip ***Scanner***, iar rezultatul o sa fie afisat folosind fluxul standard de iesire. Citirea se va face intr-o metoda ***myRead*** in care, pe langa citire, se introduce fiecare numar pozitiv intr-un obiect de tip ***Vector*** (data a clasei).

Pentru determinarea elementului maxim folositi metoda ***max*** din clasa ***Collections***!

Definiti o exceptie cu numele ***NumarNegativ*** care va afisa mesajul *"`Numarul introdus este negativ!`"*. Aceasta exceptie este aruncata in cadrul metodei de citire, atunci cand se citeste un numar negativ, si este tratata in metoda ***main***, acolo unde este apelata metoda de citire.

Problema 4

In arhiva laboratorului, gasiti clasa ***Warrior*** care modeleaza un personaj de tip razboinic si clasa ***WarriorPack*** care descrie un grup de razboinici si cat de multe daune pot produce ei. Codul acestor clase nu este unul orientat obiect si nu ofera posibilitatea de incapsulare a datelor. Rescrieti acest cod astfel incat sa utilizati mostenirea pentru a reprezenta diferitele tipuri de extraterestri, renuntand la membrul type, si sa oferiti posibilitatea de incapsulare a datelor. De asemenea, implementati o metoda ***getDamage*** in fiecare clasa derivata care sa returneze valoarea daunelor ce pot fi produse de tipul respectiv.

Adaugati metoda ***toString*** atat in clasa ***Warrior*** (tipul clasei, name, health) cat si in clasa ***WarriorPack***.

La final, rescrieti metoda ***calculateDamage***, utilizand metoda definita anterior, ***getDamage***, si testati functionalitatea codului. Afisati si datele obiectului de tip ***WarriorPack***.

poo/laboratoare/05.txt · Last modified: 2018/10/22 13:40 by carmen.odubasteanu